A Heterogeneous Computing Approach to Simulation of the Heston Stochastic Volatility Model

Kenneth A. Lindsay[1] and David J. Warne[2,3]

[1]School of Economics and Finance (QUT)
[2]High Performance Computing and Research Support (QUT)
[3]School of Mathematical Sciences (QUT)

December 6, 2015

Outline

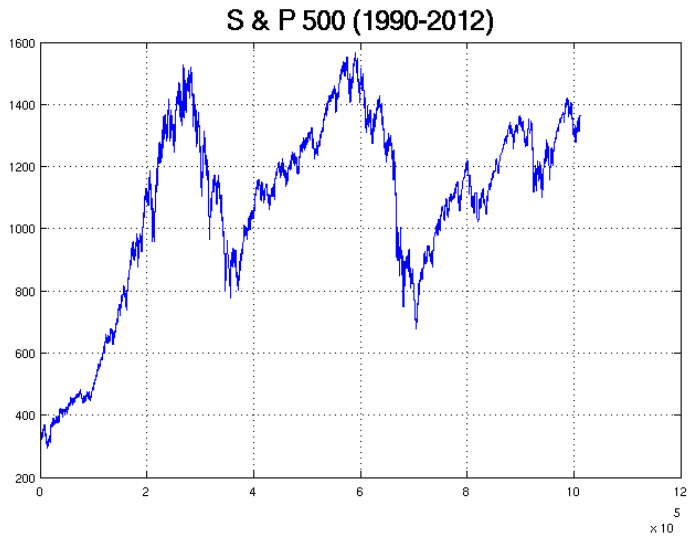
1. Introduction
   - Asset Derivative Pricing
   - Stochastic Volatility Model
2. Parameter Estimation
   - Log-likelihood
   - Particle Filtering
   - Calculation of Option Prices
3. Heterogeneous Computing Implementation
   - The Open Computing Language
   - Algorithm Parallelism
4. Performance
   - Runtime Comparison
   - Theoretical Performance
   - Efficiency and Power Comparison
5. Conclusion

Asset Derivative Pricing

- Financial index analysis
  - Fitting a mathematical model to index data to investigate properties and determine expected returns.
  - Time-series data can be more accurately fitted by inclusion of derivatives.
- What are derivatives?
  - Derive value from performance of a financial asset.
  - e.g., forwards, futures, options and swaps.
- European Options
  - A contract, purchased at a premium, for the *Right* to buy/sell a specified asset at a fixed strike price K.
  - Can only be exercise on their maturity date T (and not before).
  - Call option (*Right-to-buy*), Put option (*Right-to-sell*).

S & P 500 Index

- S & 500
  - Based on market capitalisations of 500 American companies.
  - Composite of various classes of assets.
  - Leading benchmark in overall American stock market.
  - Investors can purchase options based on this index.



- Dataset
  - Daily S&P index value from 1990 - 2012.
  - Includes options data.

Heston's Stochastic Volatility Model

Given independent Wiener processes $W_1(t)$ and $W_2(t)$, the index level $S(t)$ and volatility $V(t)$ are modelled using the SDE's [1].

Heston Model (Physical Measure)

$$\frac{dS}{S} = (r - q - \xi_S V)\,dt + \sqrt{V}\left(\sqrt{1-\rho^2}dW_1 + \rho dW_2\right)$$
$$dV = \kappa_P(\gamma_P - V)\,dt + \sigma\sqrt{V}dW_2$$

Here r is the risk-free interest rate, q is the dividend-price ratio, ξ_S is the equity premium, γ_P is the long-time mean volatility, κ_P is the rate that V reverts to γ_P, and σ is the volatility of volatility.

Heston's Stochastic Volatility Model

To determine the value of an option the following variation is used,

Heston Model (Risk-Free Measure)

$$\frac{dS}{S} = (r-q)\,dt + \sqrt{V}\left(\sqrt{1-\rho^2}dW_1 + \rho dW_2\right)$$
$$dV = \kappa_Q\left(\gamma_Q - V\right)dt + \sigma\sqrt{V}dW_2$$

Here $\kappa_Q = \kappa_P + \lambda\sigma^2$ and $\gamma_Q = \frac{\kappa_P\gamma_P}{\kappa_Q}$. λ is a value which is to be determined from data.

Likelihood Maximisation

Let $X_0, \ldots, X_T$ be observations of the index at times $t_0, \ldots, t_T$. Given model likelihood $\mathcal{L}(\theta; X_0, \ldots, X_T)$, we can perform parameter estimation by solving the following problem.

Maximum Likelihood Estimator (MLE)

$$\hat{\theta} = \arg\max_{\theta} \hat{l}\left(\theta; X_0, \ldots, X_T\right), \tag{1}$$

where,

$$\hat{l}\left(\theta; X_0, \ldots, X_T\right) = \frac{1}{T} \ln \mathcal{L}\left(\theta, X_0, \ldots, X_T\right) \tag{2}$$

Log-likelihood

For index data $X_k = \left(S_k, V_k, H_k^1, \ldots, H_k^M\right)$, where H_k^j is the price of the j-th option at t_k.

Log-likelihood

$$\hat{l}\left(\theta; X_0, \ldots, X_T\right) = \frac{1}{T}\left[\sum_{j=1}^{M} \ln g\left(H_0^j \middle| \tilde{H}_0^j; \theta\right) + \sum_{k=1}^{T}\left(\ln f_P\left(X_k, \Delta t_k | X_{k-1}; \theta\right) + \sum_{j=1}^{M} \ln g\left(H_k^j \middle| \tilde{H}_k^j; \theta\right)\right)\right],$$

where $\tilde{H}^j$'s are model predicted option prices, g is the distribution of option pricing errors and f_p is the transitional density of the physical model.

Recursive Filtering

Generally the volatilities V_k are not directly observed. Thus f_p is not sufficient as V_k must be filled in for each time step t_k.

- First assume we have $f\left(V_{k-1}|Z_{k-1}\right)$, with $Z_n = \{X_j\}_{j=0}^{j=n}$.
- Then we can derive $f\left(V_k|Z_k\right)$ by evaluating the integrals,

$$f\left(X_k, V_k|Z_{k-1}\right) = \int_{\mathcal{V}} f\left(X_k|V_k, V_{k-1}\right) f\left(V_k|V_{k-1}\right) f\left(V_{k-1}|Z_{k-1}\right) dV_{k-1},$$

$$f\left(X_k|Z_{k-1}\right) = \int_{\mathcal{V}} f\left(X_k, V_k|Z_{k-1}\right) dV_k.$$

Now apply Bayes' Theorem,

$$f\left(V_k|Z_k\right) = \frac{f\left(X_k, V_k|Z_{k-1}\right)}{f\left(X_k|Z_{k-1}\right)}$$

Recursive Filtering

The integrals for $f(X_k, V_k|Z_{k-1})$ and $f(X_k|Z_{k-1})$ are computed numerically using Monte Carlo integration,

1. Given N particles sampled from $f(V_{k-1}|Z_{k-1})$,
2. Simulate the physical Heston Model forward to t_k (using Euler discretisation); this provides $f(V_k|V_{k-1})$.
3. Evaluate $f(X_k|V_k, V_{k-1})$ for each particle. This step is the computational part as it requires evaluation of the model option prices .
4. Let L_j be the value of $f(X_k|V_k, V_{k-1})$ for particle j. Then $f(X_k|Z_{k-1}) = \sum_{j=1}^{N} L_j$.

Expected Payoff (Call Option)

The risk-neutral Heston Model can is used to compute option prices,

Call Option Payoff

Given asset spot price S_0 and $Y = \ln S/S_0$, the expected payoff for a call option with maturity T and strike price K is,

$$\tilde{H} = S_0 \int_{\ln \xi}^{\infty} (e^y - \xi) \int_{-\infty}^{\infty} f_Q(0, V, T, |y, v; \theta)\, dydv,$$

where $\xi = K/S_0$ and f_Q is the transitional density function of the risk-neutral Heston Model.

The Backward Kolmogorov Equation

Let $Y = \ln S/S_0$ and apply Itō's Lemma to the risk-neutral Heston Model,

$$dY = \left((r-q) - \frac{V}{2}\right)dt + \sqrt{V}\left(\sqrt{1-\rho^2}dW_1 + \rho dW_2\right)$$
$$dV = \kappa_Q\left(\gamma_Q - V\right)dt + \sigma\sqrt{V}dW_2$$

It can be shown that the Backward Kolmogorov Equation (BKE) for this stochastic diffusion process is,

$$\frac{\partial f_Q}{\partial t} = -\left(r - q - \frac{V}{2}\right)\frac{\partial f_Q}{\partial Y} - \kappa_Q\left(\gamma_Q - V\right)\frac{\partial f_Q}{\partial V}$$
$$- \frac{V}{2}\left[\frac{\partial^2 f_Q}{\partial Y^2} + 2\rho\frac{\partial^2 f_Q}{\partial Y \partial V} + \sigma^2\frac{\partial^2 f_Q}{\partial V^2}\right]$$

Characteristic Equation

We can transform the BKE using the Characteristic Equation of f_Q,

Characteristic Equation

$$F_Q(Y, V, t, \omega_y, \omega_v; \theta) = \iint_{\mathbb{R}^2} f_Q(Y, V, t|y, v; \theta)\, e^{i(\omega_y y + \omega_v v)} dydv$$

Using the transformed BKE it is possible to obtain a semi-closed form solution to F_Q,

$$F_Q(Y, V, t, \omega_y, \omega_v) = e^{B_0(\tau,\omega_y,\omega_v) + B_1(\tau,\omega_y,\omega_v)Y + B_2(\tau,\omega_y,\omega_v)V},$$

with $\tau = T - t$, and B_0,B_1, and B_2 are solutions to a set of ODE's which must be numerically solved.

Fourier coefficients

Under the assumption that f_Q has compact support $[-\beta, \beta]$ in y, then the following approximation can be applied [2, 3].

Approximating Fourier Series

$$\int_{-\infty}^{\infty} f_Q\left(Y, V, T | y, v; \theta\right) dv = \sum_{k=-\infty}^{\infty} c_k e^{-\frac{k\pi i y}{\beta}},$$

with coefficients,

$$c_k \approx \frac{1}{2\beta} F_Q\left(Y, V, T, \omega_k, 0; \theta\right), \ \omega_k = \frac{k\pi}{\beta}.$$

This enables the option prices to be evaluated and the particle filter is complete.

The Open Computing Language

- Previous work has been focused on GPUs [2].
  - Specifically NVidia GPUs.
  - Programmed in CUDA C.
- Now we look to a more general approach,
  - The parallelism inherent in our methods lend themselves to many architectures.
  - GPUs (not just NVidia), Multi-core processors, and Xeon Phi's.
- The complete heterogeneous approach.
  - Implementation in OpenCL.
  - Code written once (mostly), with many available targets (including future targets like FPGAs).
  - Enables more realistic benchmarks between devices.
  - Common misconception: GPUs should not give 100x speedup.

The Open Computing Language

- The Execution Model:
  - A serial "host" processor.
  - Multiple parallel devices,
    - Asynchronous Compute Units.
    - Processing Elements executing in SIMD.
- The Programming Model:
  - Standard sequential model for host code.
  - Device code specifies:
    - The serial logic of a single *work item* to be executed on a processing element.
    - Communication between work items running on the sample compute unit.
    - Work items to be executed on the sample compute unit are called a *Work Group*.
- The mapping for compute units and processing elements to the physical hardware is vendor specific.

The Open Computing Language

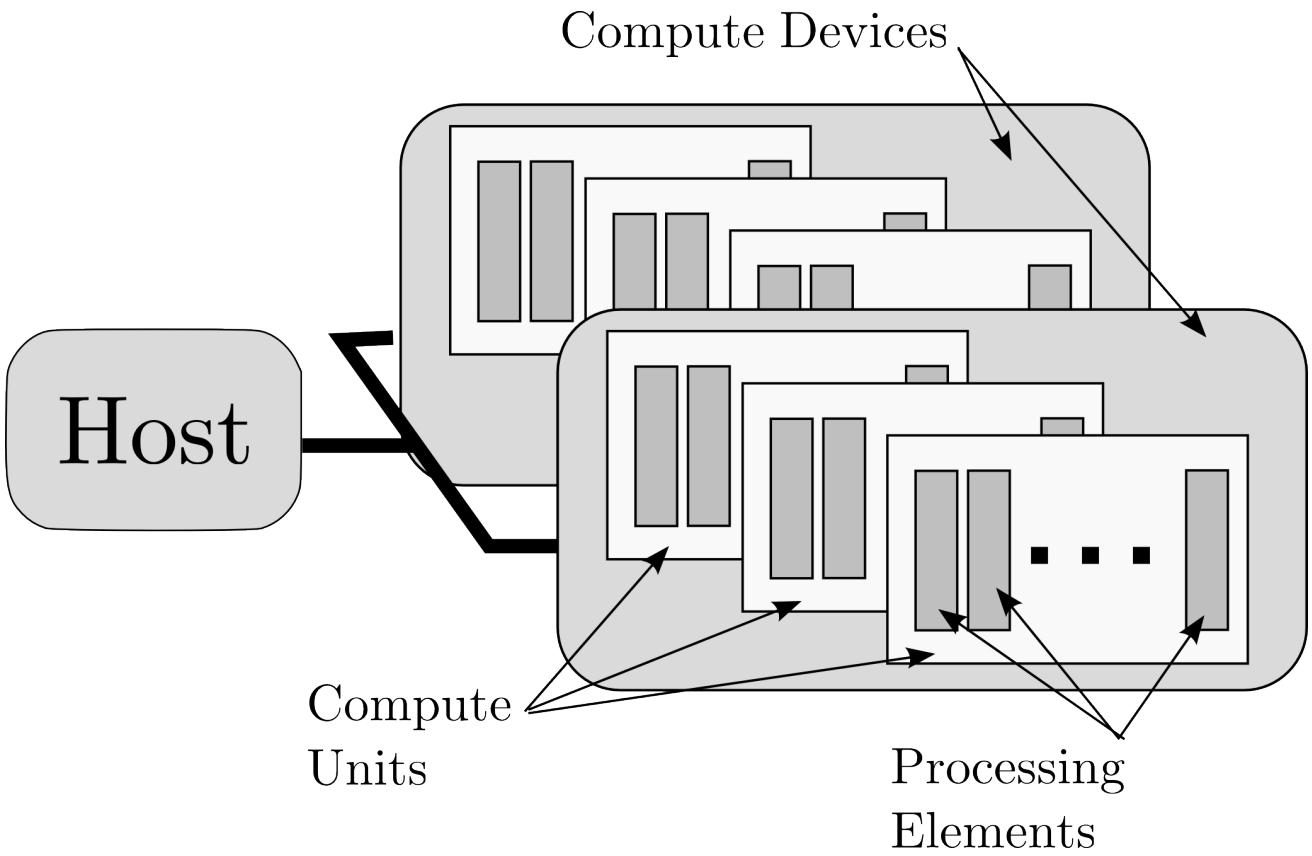
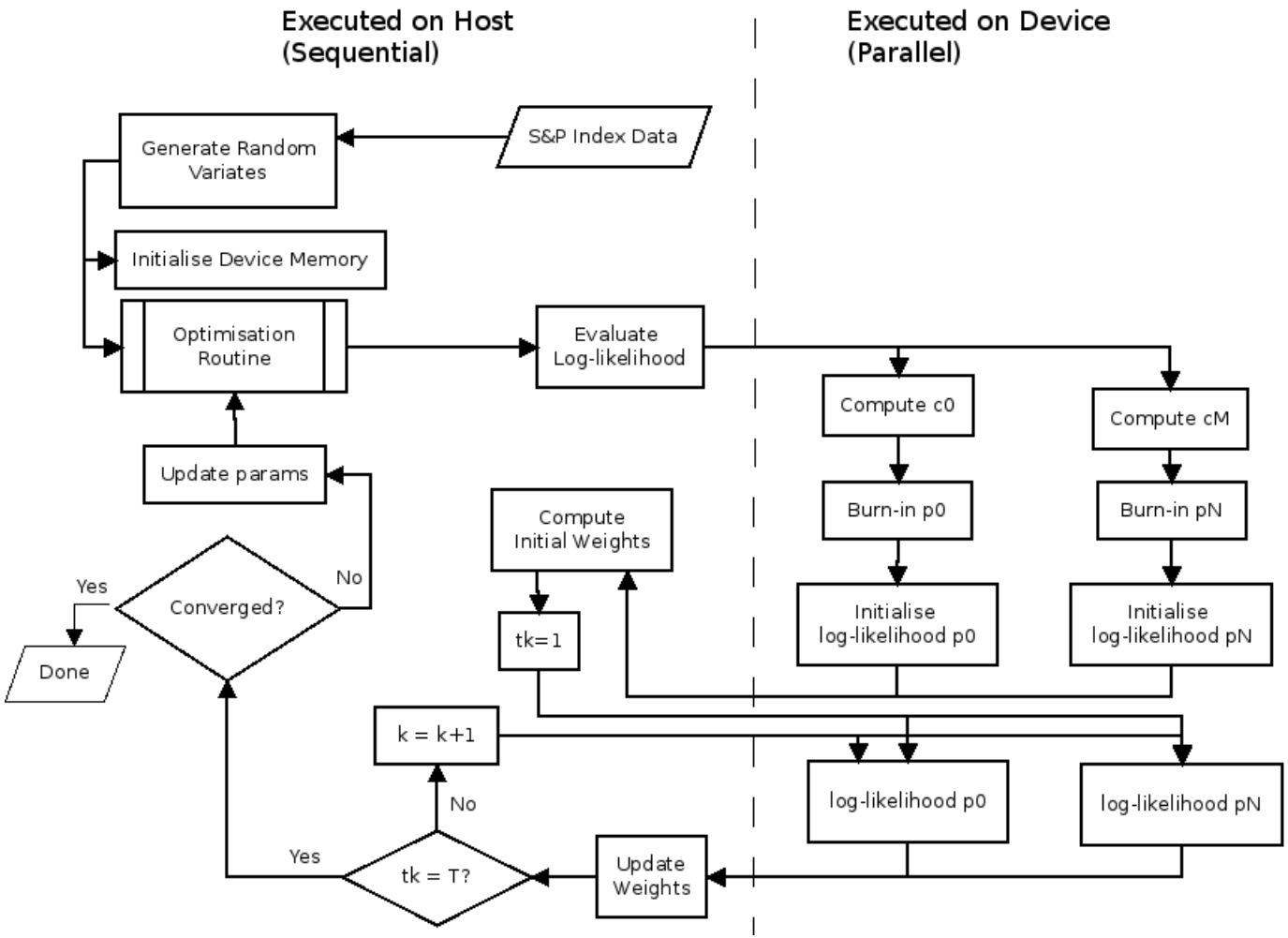


Figure: The execution model.

Parallel implementation of particle filter

- The particle filter method can easily exploit parallelism.
  - Each forward step from t_{k-1} to t_k is embarrassingly parallel.
  - Particles act independently.
- Dependency between forward steps.
  - Information from all particle likelihood contributions is used to construct new weights
  - The new particle volatilities and weights are used for the new forward step.
- Fourier coefficients need to be computed.
  - Performed at the start of each likelihood evaluation.
  - Each frequency ω_k can be computed independently.
  - This involves solving the ODEs for B_0, B_1, B_2 and evaluating F_Q.

Algorithm


Executed on Host
(Sequential)
Executed on Device
(Parallel)
S&P Index Data
Generate Random Variates
Initialise Device Memory
Optimisation Routine
Evaluate Log-likelihood
Compute c0
Compute cM
Burn-in p0
Burn-in pN
Initialise log-likelihood p0
Initialise log-likelihood pN
Update params
Compute Initial Weights
tk=1
Converged?
Yes
No
Done
k = k+1
log-likelihood p0
log-likelihood pN
No
Yes
tk = T?
Update Weights

Initial Performance Comparison

- Our OpenCL Implementation is compatible with the Intel and NVidia OpenCL.
- Comparion against CUDA C version for Tesla GPU,
  - CUDA C $\approx$ 48 (mins) (512 blocks $\times$64 threads)
  - OpenCL $\approx$ 51 (mins) (512 Work Groups $\times$64 Work Items)
- The Optimum distribution of work items is device dependent.
  - Executed code for 32768 particles using CPU, GPU, and Xeon Phi.
  - Selected different blocking partitions.

Run times

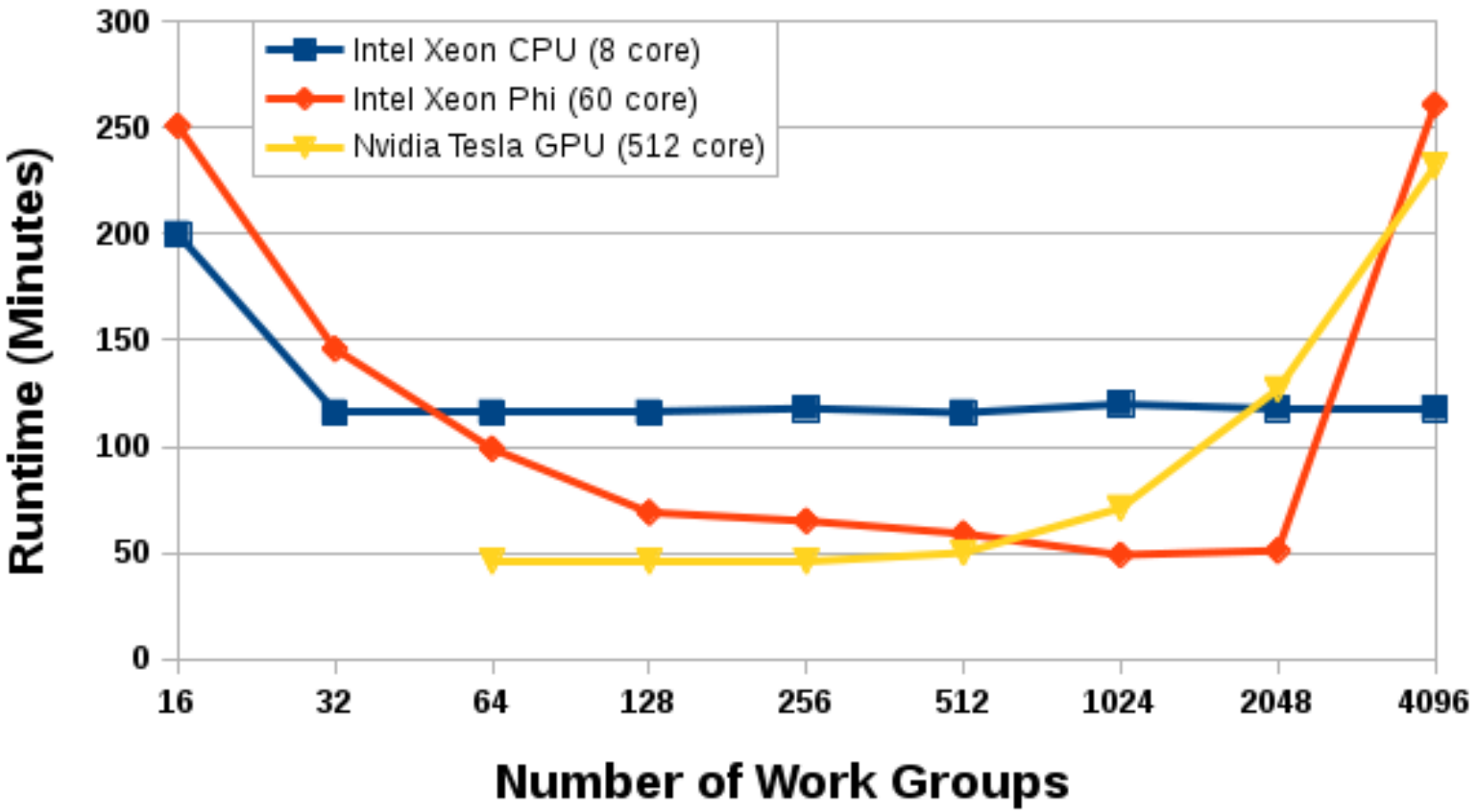


Figure: Performance tuning for $32,768$ particles.

Computational Complexity

- The major computational burden is the evaluation of the log-likelihood.
- The number of double precision floating point operations (FLOPs) required

$$C_L = 1 + 2C_{div} + 7C_{exp} + C_F + C_S + C_I + N_t C_{PF},$$

with

$$\begin{aligned} C_F &= 1 + 544 D_m, \\ C_S &= 5 + C_{div} + C_{sqrt} + N_b (9 + C_{sqrt}) \\ C_{PF} &= N_p (2 + C_{div} + C_{sqrt} + C_{ln} + N_s (15 + 2C_{sqrt})) \\ &\quad + N_p (N_o (23 + 4C_{div} + 3C_{ln} + C_{exp} \\ &\quad + N_f (30 + 2C_{div} + C_{exp} + 4C_{trig}))) \end{aligned}$$

Theoretical Run times

- We can compute rough FLOP count.
  - For experiments $N_t = 4,538$, $N_p = 32,768$, $N_s = 4$, $N_o = 4$, $N_b = 3,024$, and $N_f = 200$, $D_m = 90$.
  - Roughly $C_{div} \approx 2$, $C_{sqrt} \approx 3$, $C_{exp} \approx C_{ln} \approx C_{trig} \approx 120$.
  - All together yields $C_L \approx 7.6 \times 10^{13}$ FLOPs. The function gets executed 11 times on average.
- Our OpenCL Code was executed on three equivalent generation architectures
  - Intel E5-2670 CPU ($R_{max} = 166$ GFLOPs/sec)
  - NVidia M2090 Tesla GPU ($R_{max} = 666$ GFLOPs/sec)
  - Intel Xeon Phi 5110P co-processor ($R_{max} = 1$ TFLOP/sec).

Theoretical Run times

- We can now compute theoretcial (optimistic) run times,
  - Intel E5-2670 CPU: ≈ 83 (mins).
  - NVidia M2090 GPU: ≈ 21 (mins) .
  - Intel Xeon Phi 5110P: ≈ 14 (mins) .
- Actual Peak run times,
  - Intel E5-2670 CPU: ≈ 116 (mins).
  - NVidia M2090 GPU: ≈ 46 (mins) .
  - Intel Xeon Phi 5110P: ≈ 49 (mins) .

Efficiency

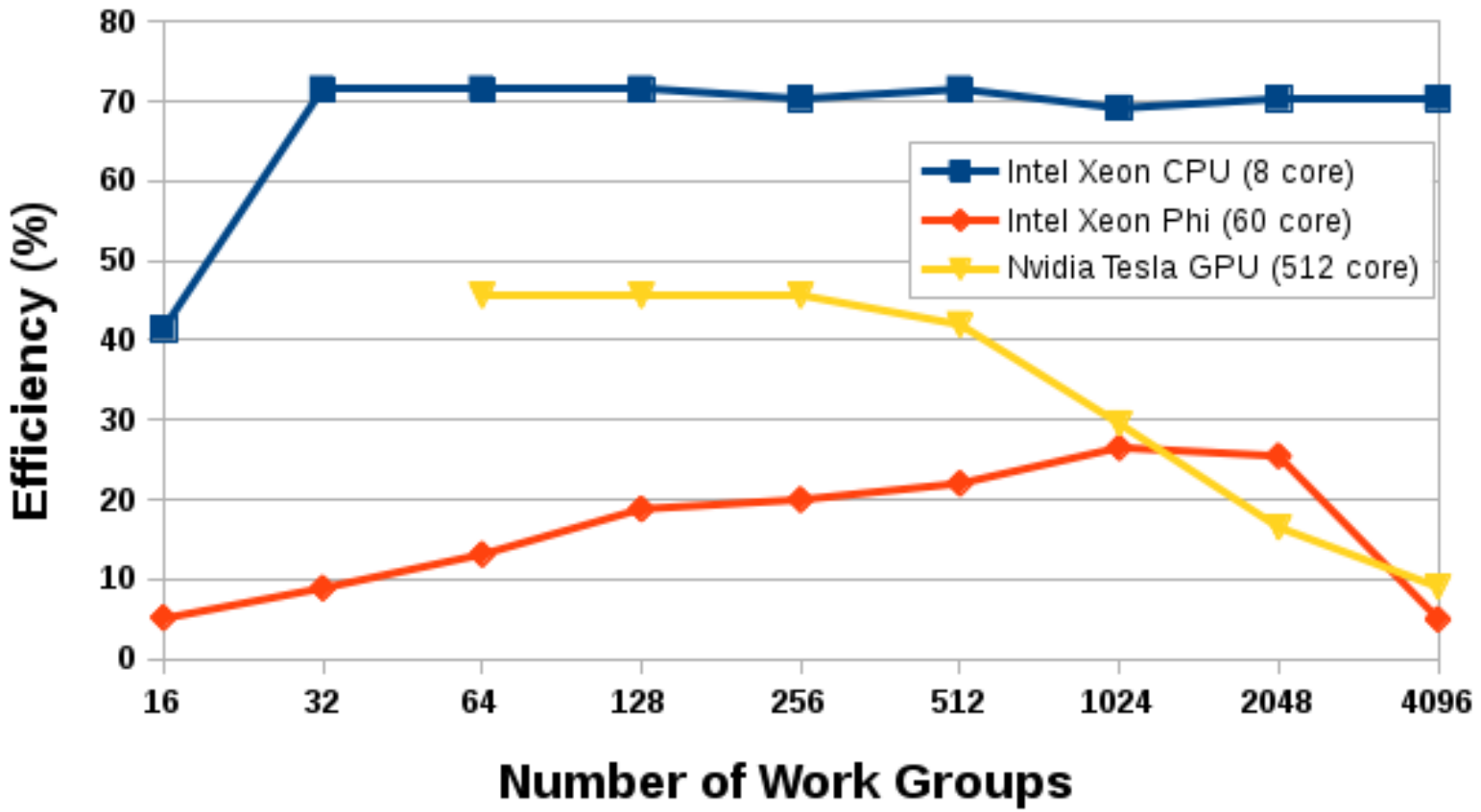


Figure: Efficiency for different work group sizes.

Power Consumption

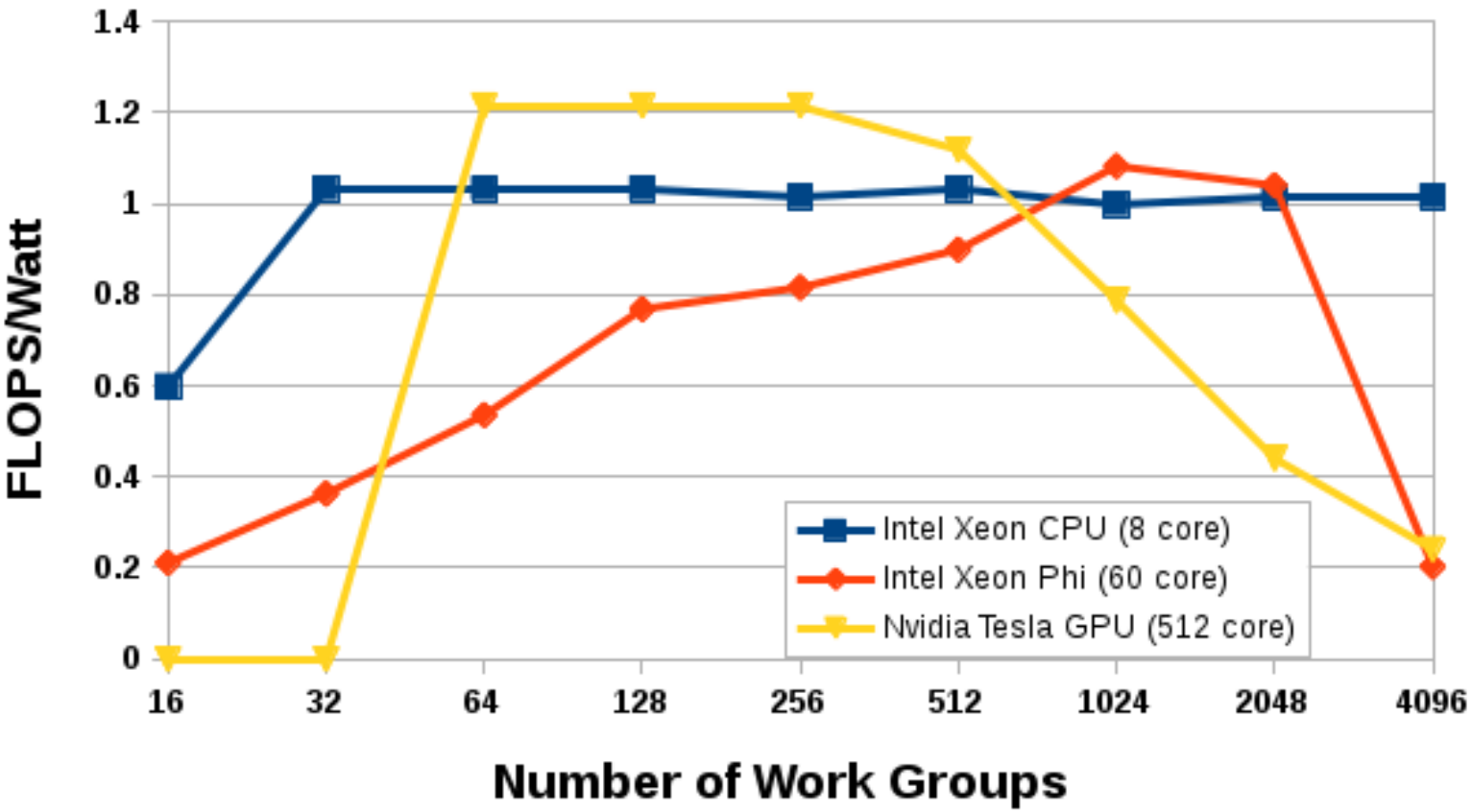


Figure: FLOPs/Watt for different work group sizes.

Conclusion

- Parameter estimation of stochastic volatility models are computationally expensive.
  - Particle filtering method enables parallelism.
  - Initial implementation in CUDA C provided speedup over serial code.
- Motivation to re-write in OpenCL,
  - Automatic portability to multiple devices.
  - Fair comparison of devices.
- Findings
  - OpenCL and CUDA C nearly identical performance for same running conditions.
  - Phi's quite disappointing, possibly due to very early OpenCL implementation.
- Complexity analysis provides more insight
  - OpenCL for CPU closer to theoretical.
  - The Phis and GPU are marginally better for Power.

Thank You!

[1] S. L. Heston.
A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options.
Review of Financial Studies, 327–343, 1993.

[2] A. S. Hurn, K. A. Lindsay, and A. J. McClelland.
Estimating the parameters of stochastic volatility models using option price data.
Journal of Business and Economic Statistics, 33(4):579–594, 2014.

[3] A. S. Hurn, K. A. Lindsay, and A. J. McClelland.
On the Efficacy of Fourier Series Approximations for Pricing European and Digital Options.
Applied Mathematics, 5(17):2786–2807, 2015.